Midterm 3 Study Guide

The test covers all the material discussed in sections 9.4-9.7, the CFG handout (sections 5.1 and 5.2.1), and the pushdown-automata[1] and parsing[2] notes, and homeworks 9 through 12. The following set of questions is meant to help guide your study and is not meant to be exhaustive of all the possibilities.

1. What are the various building blocks for (pure) regular expressions? Provide regular expressions for the following languages on the alphabet $\{a, b\}$:

   - All strings that start with three a's and have length 5.
   - All strings that start with three a's and have length at most 5.
   - All strings that start with a and end in b, or start with b and end in a.

2. State what the pumping lemma for regular languages says. Briefly explain why it is true.

3. Describe how a pushdown automaton differs from a finite automaton, and what its (non-deterministic) state transitions look like. Explain in simple terms why the PDAs have more computational power than NFAs/DFAs.

4. For the following non-regular languages: prove that they are not regular, build PDAs for them, build context-free grammars for them, and demonstrate the PDA execution as well as the CFG derivation for the provided input strings:

   - $\{x^n y^n \mid n \geq 0\}$, $s = x^3 y^3$.
   - $\{x^n y^{2n} \mid n \geq 0\}$, $s = x^2 y^4$.
   - $\{x^{2n} y^n \mid n \geq 0\}$, $s = x^4 y^2$.
   - $\{x^n z^m y^n \mid n, m \geq 0\}$, $s = x^2 z y^2$.
   - $\{x^n \mid n \geq 0\} \cup \{x^n y^n \mid n \geq 0\}$, $s = x^2 z y^2$.

5. Explain why the union, intersection, concatenation, and Kleene star of regular languages is regular, and the same for context-free languages.

6. Describe the two different PDAs corresponding to a given CFG. Follow the derivation of a particular string on the PDAs, showing the evolution of the stack and the process of the input. Recall that one of the PDAs builds a leftmost derivation while the other builds a rightmost derivation.

7. Build the item-set DFAs for the following grammars:

   ```
   S -> eps | a | b | aSa | bSb
   ```

   and:

   ```
   E -> a | (A)
   A -> E | E A
   ```

[1] notes/new_cfg_pushdown.html
[2] notes/parsing.html